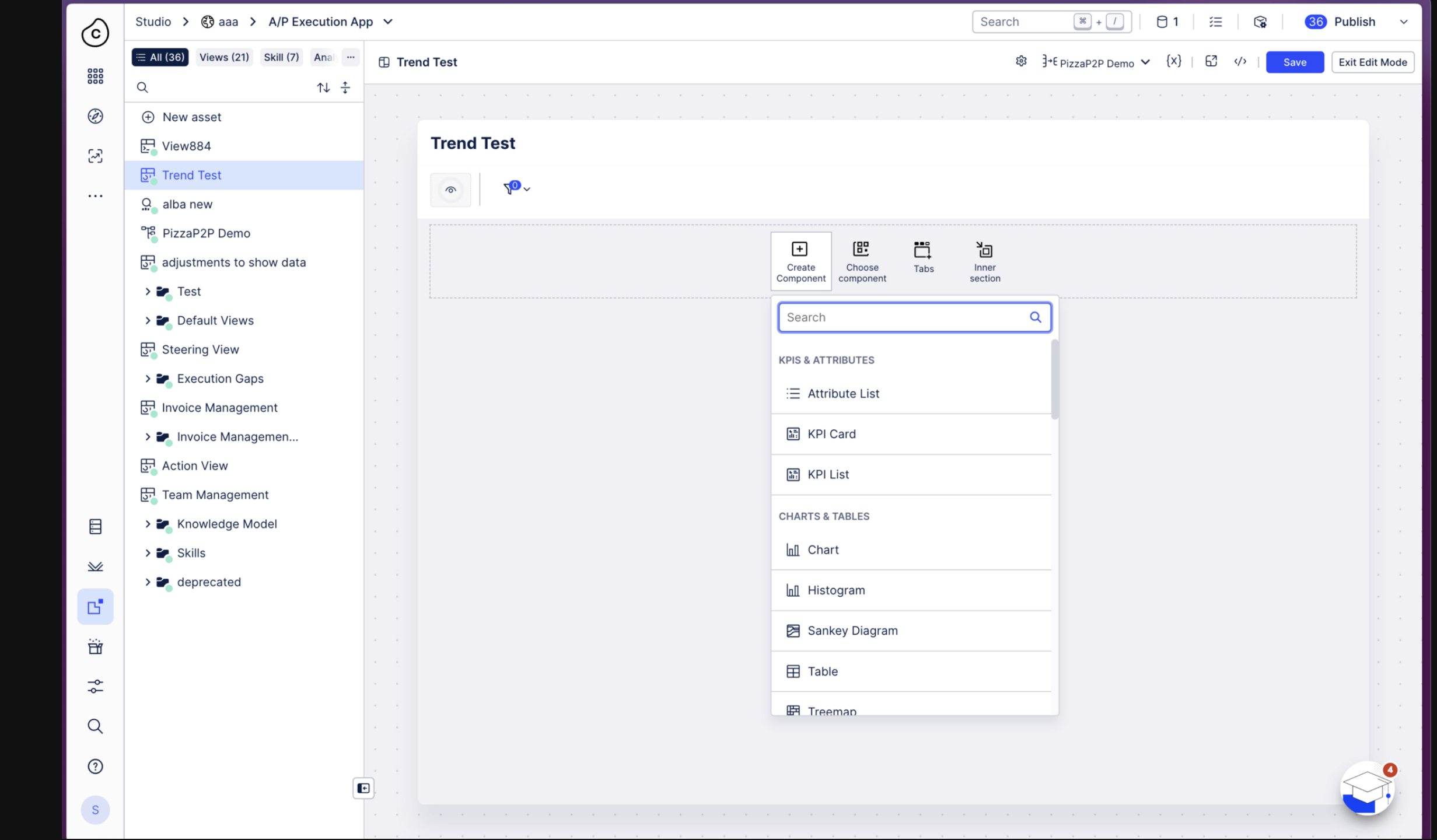
How to Ship Updates to 40+ Apps Every Week With Nx

— by Santosh Yadav

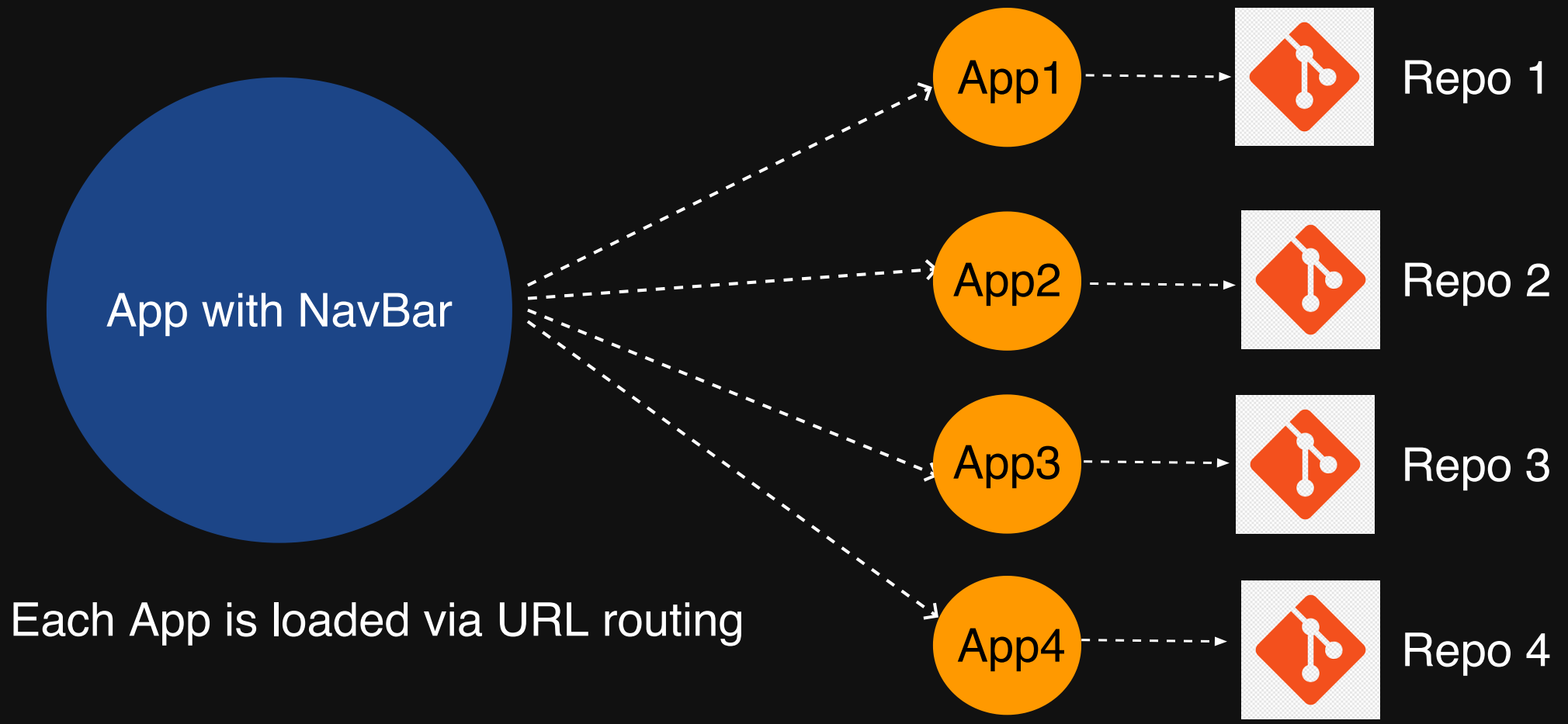
Our Studio Application


Studio
aaa
A/P Execution App
Search
1
36 Publish
All (36)
Views (21)
Skill (7)
Trend Test
PizzaP2P Demo
Save
Exit Edit Mode
New asset
View884
Trend Test
alba new
PizzaP2P Demo
adjustments to show data
Test
Default Views
Steering View
Execution Gaps
Invoice Management
Invoice Managemen...
Action View
Team Management
Knowledge Model
Skills
deprecated
Trend Test
Create Component
Choose component
Tabs
Inner section
Search
KPIS & ATTRIBUTES
Attribute List
KPI Card
KPI List
CHARTS & TABLES
Chart
Histogram
Sankey Diagram
Table
Treemap

Problem Statement

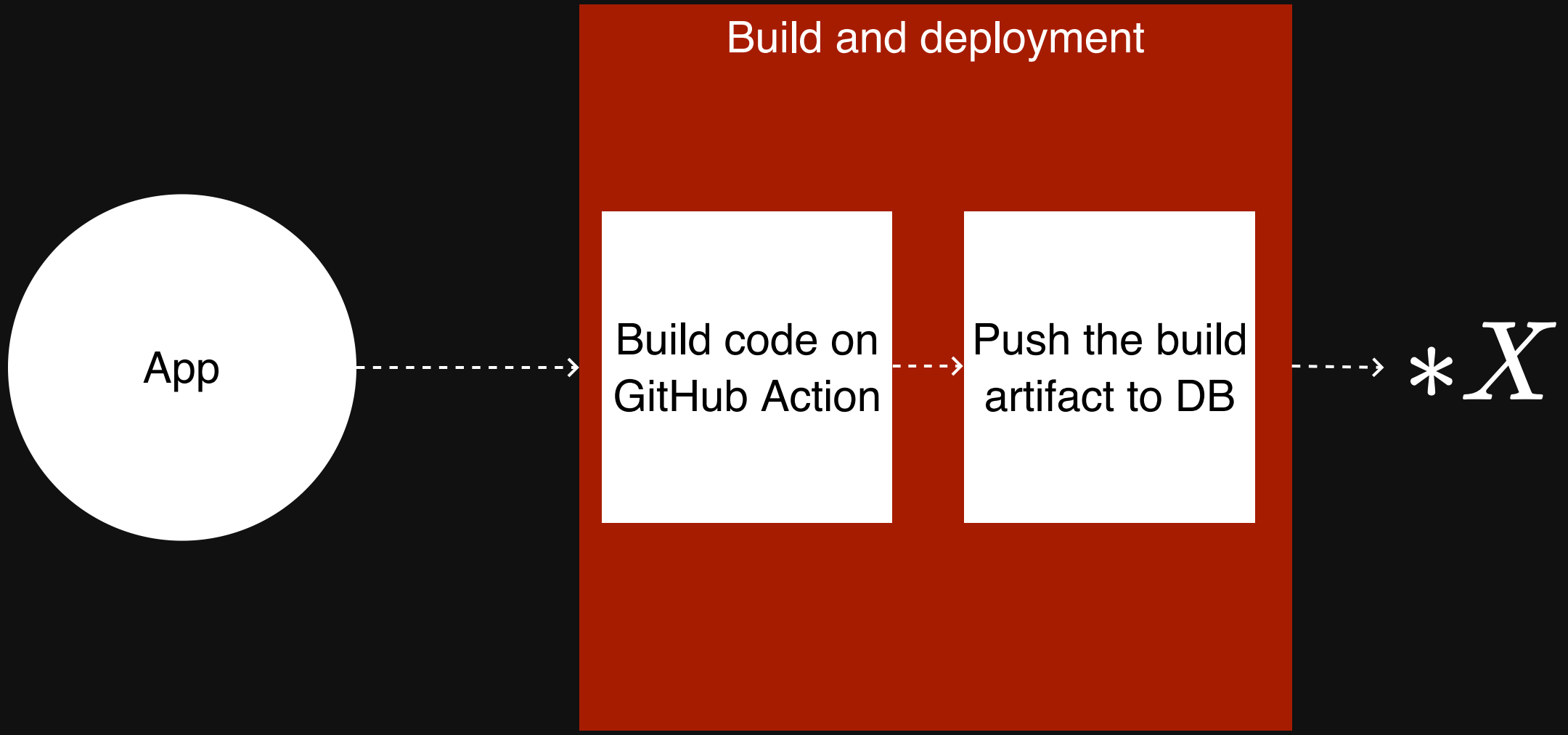
Why are we doing this?

Before module federation with Nx



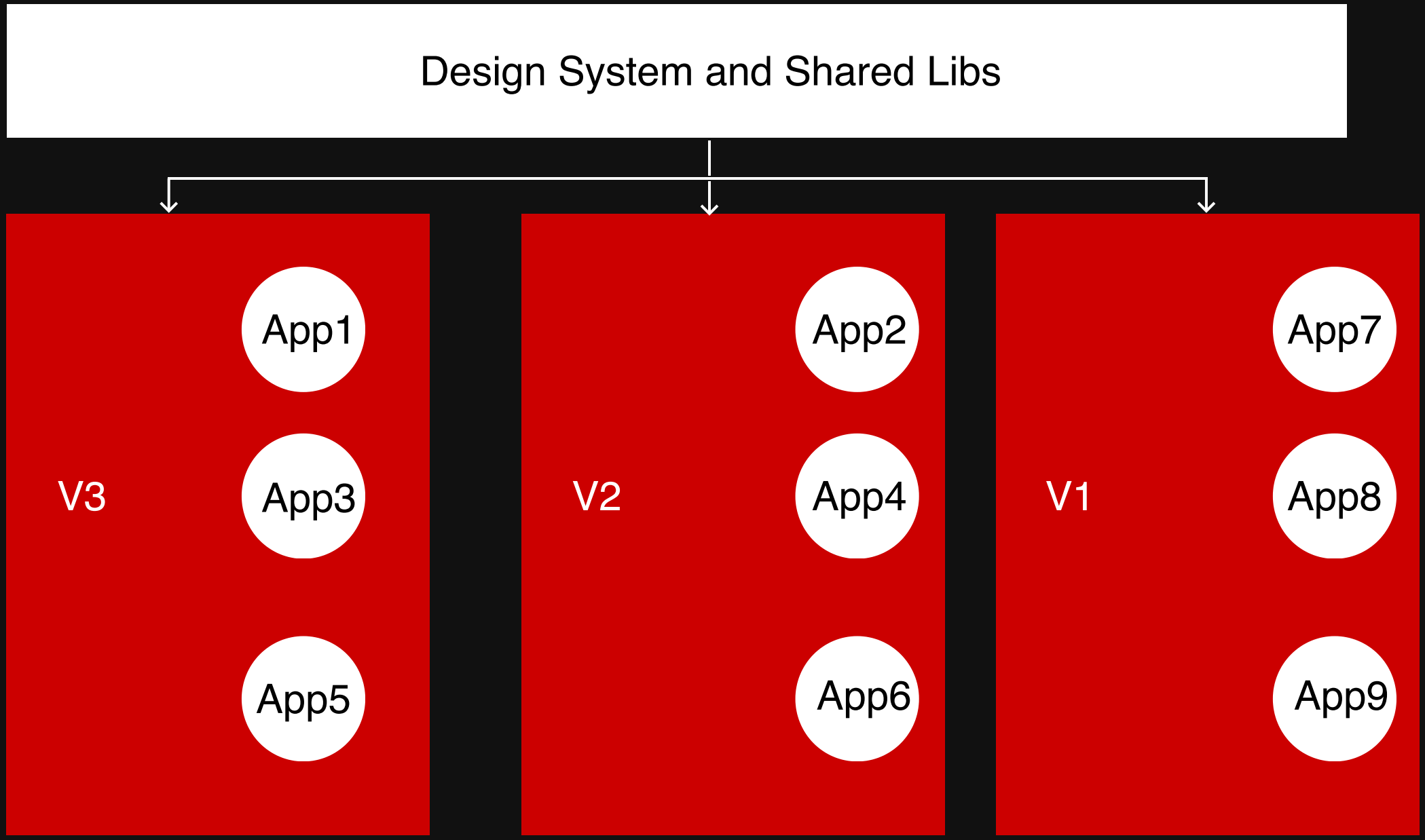
Each App is loaded via URL routing

Build and Deployment workflow for each App



X = number of Apps

Sharing code became problem



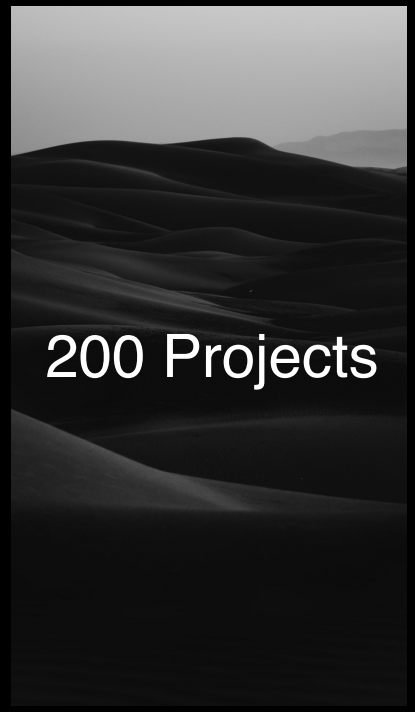

Angular
Angular
Corporate needs you to find the differences between this picture and this picture.
They’re the same picture.

- Page reloads for each app redirect
- Huge bundle size
  - No tree shaking
  - No lazy loading
- Too much effort in upgrading to the new Angular version.
- Maintaining multiple versions of shared libs and design system
- Had to synchronize releases across apps to change design system at the same time.

What is Nx

A build tool with mono-repo support

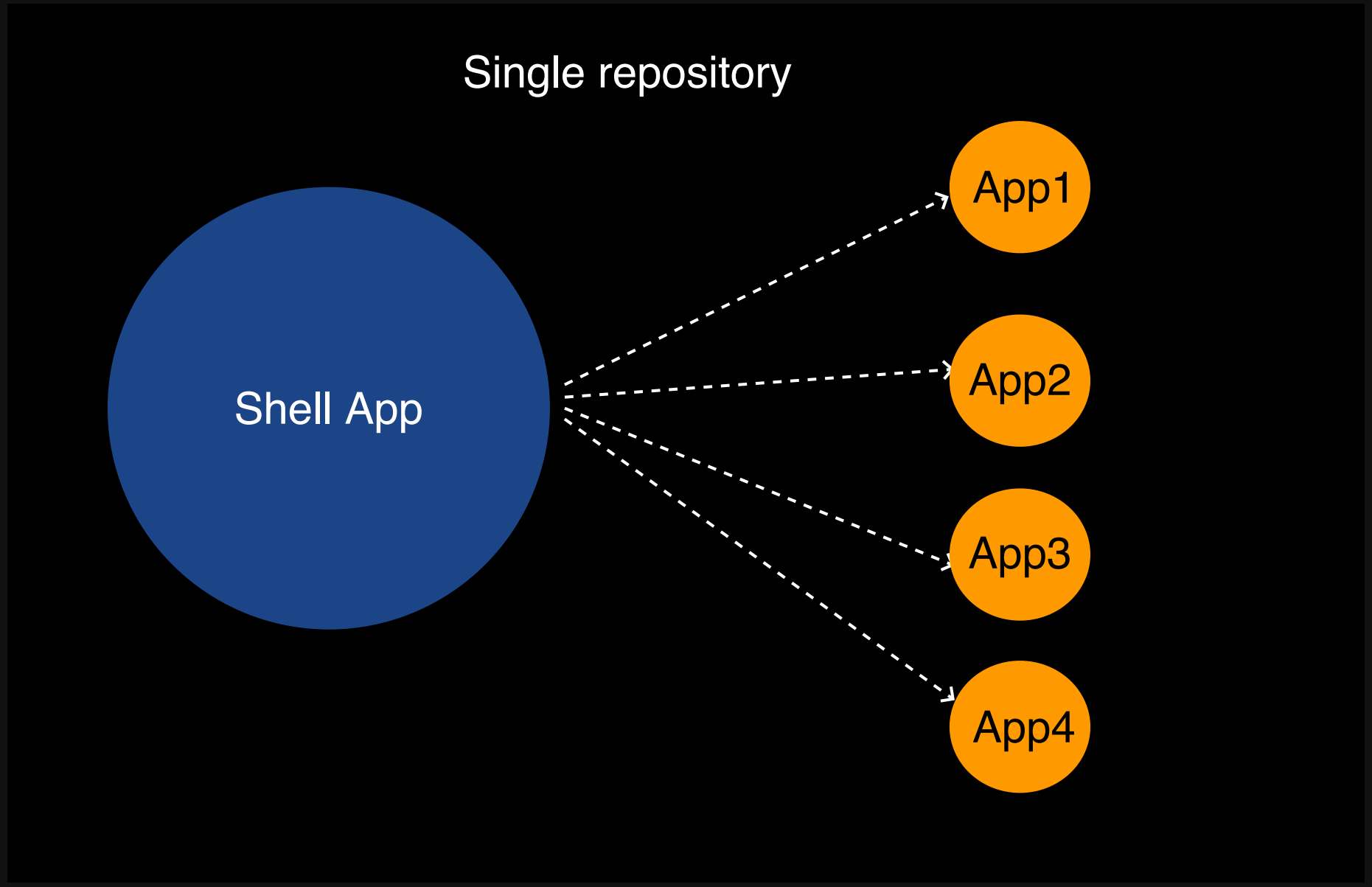
- Provides build cache for tasks like build, test
- Framework and technology agnostic
- Plugin based so you can bring your own tool
- Support major frameworks like Angular, React and Vue out of the box
- Support for micro-frontend
- Support for backend technologies like spring and .net core



Our Codebase

We have a mono-repo with **Angular** and **Nx**, which contains Angular Apps, NextJs App, Libraries, Unit Test, and e2e. We use Trunk based development approach.

With module federation with Nx



Tasks

1.

Build

Build all the projects including apps and libraries and bundle them together.

2.

Unit Test

We use jest and cypress component test to write unit tests and run them before merging our code.

3.

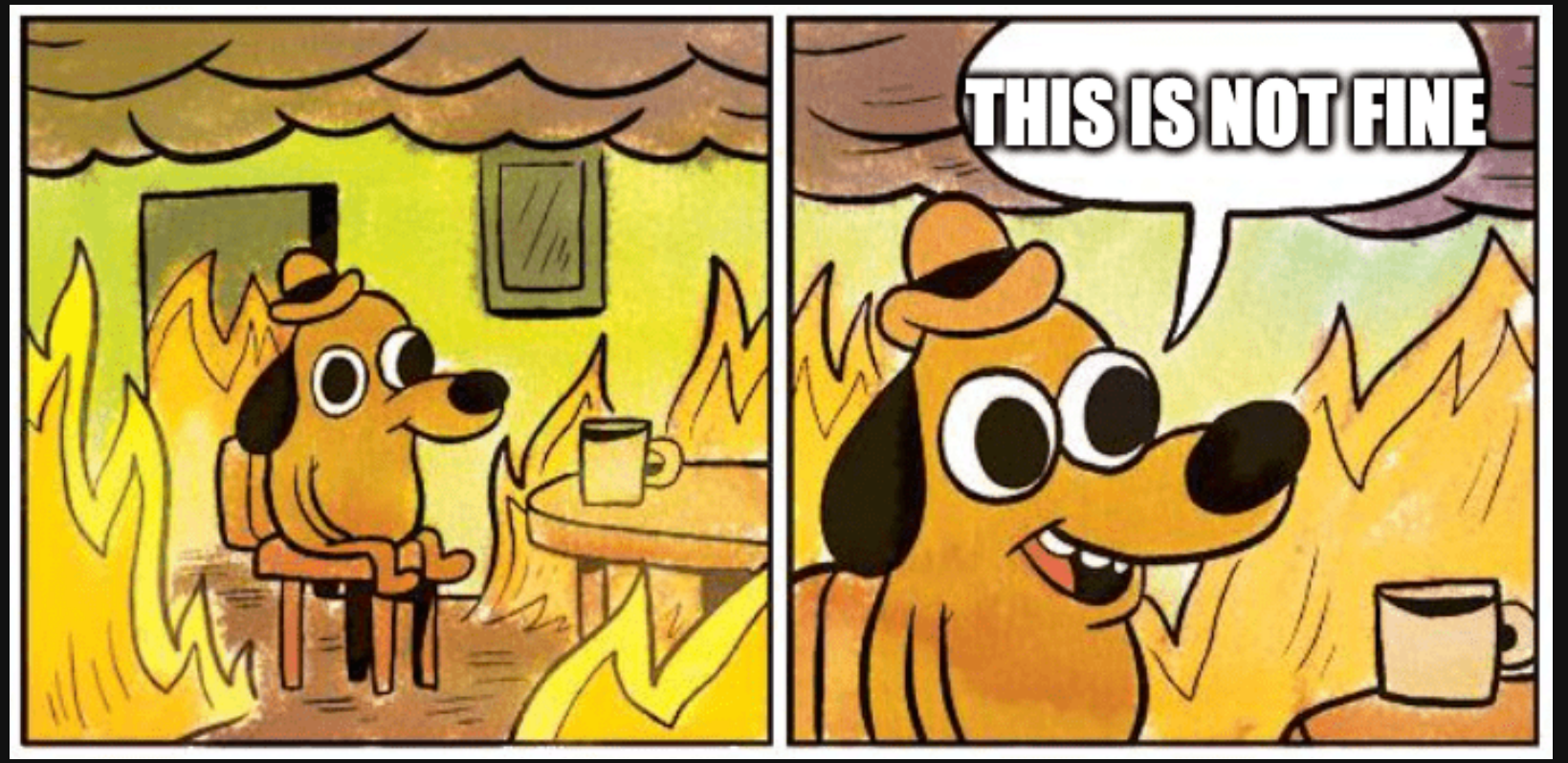
e2e

For end to end we run all User Journey test suites before going to development.

```
npx nx generate
@nx/angular:host
--name=shell
--remotes=home,about,blogs
```

Feature Flags

Feature Flag is tricky but becomes very essential when we are shipping features frequently and want to make sure we dont break existing functionality.


THIS IS NOT FINE

Shipping code without feature flag


SENORGIF.COM

- Ability to ship code based on
  - User
  - Cluster
  - % of users or customers
  - specific build
  - combination of all the above
- Isolate bugs
- Ability to rollback feature flags
- Ship features with more confidence

What tool do we use for Feature Flag?


M18 LITHIUM
COMPACT

```
nx add @nx/plugin

nx g @nx/plugin:plugin
nx-plugin
--directory infra/nx-plugin
```

Developer Experience

1.

Ability to enable/disable flag

Update feature flags by creating a Pull Request

2.

Dashboard

Dashboard to view all active features

3.

Weekly Alert

Alert to clear the feature flags once they are GA (Generally Available)


THIS IS FINE.

Proof of Concepts

We need our developers to ship POCs fast so they can experiment and meet our customer's future needs.

```
@nx/angular:remote
--name=llm-agent
--no-interactive
```


BUT
WHAT ABOUT CUSTMIZATION

Nx Generator

```
export default async function (tree: Tree, schema:
ApplicationGeneratorSchema) {

  // call built-in generator from Nx
  await generateAngularProject(tree, {
    standalone: false,
    name: schema.name,
    style: "scss",
    skipPackageJson: true,
    port: schema.port,
    e2eTestRunner: E2eTestRunner.None,
    tags: tags,
    prefix: "org-name",
  });

  // Add cypress component test when apps are created
  await generateComponentTests(tree, {
    project: schema.name,
    port: schema.componentTestPort,
    generateTests: false,
    buildTarget: `${schema.name}:build-component-test`,
```

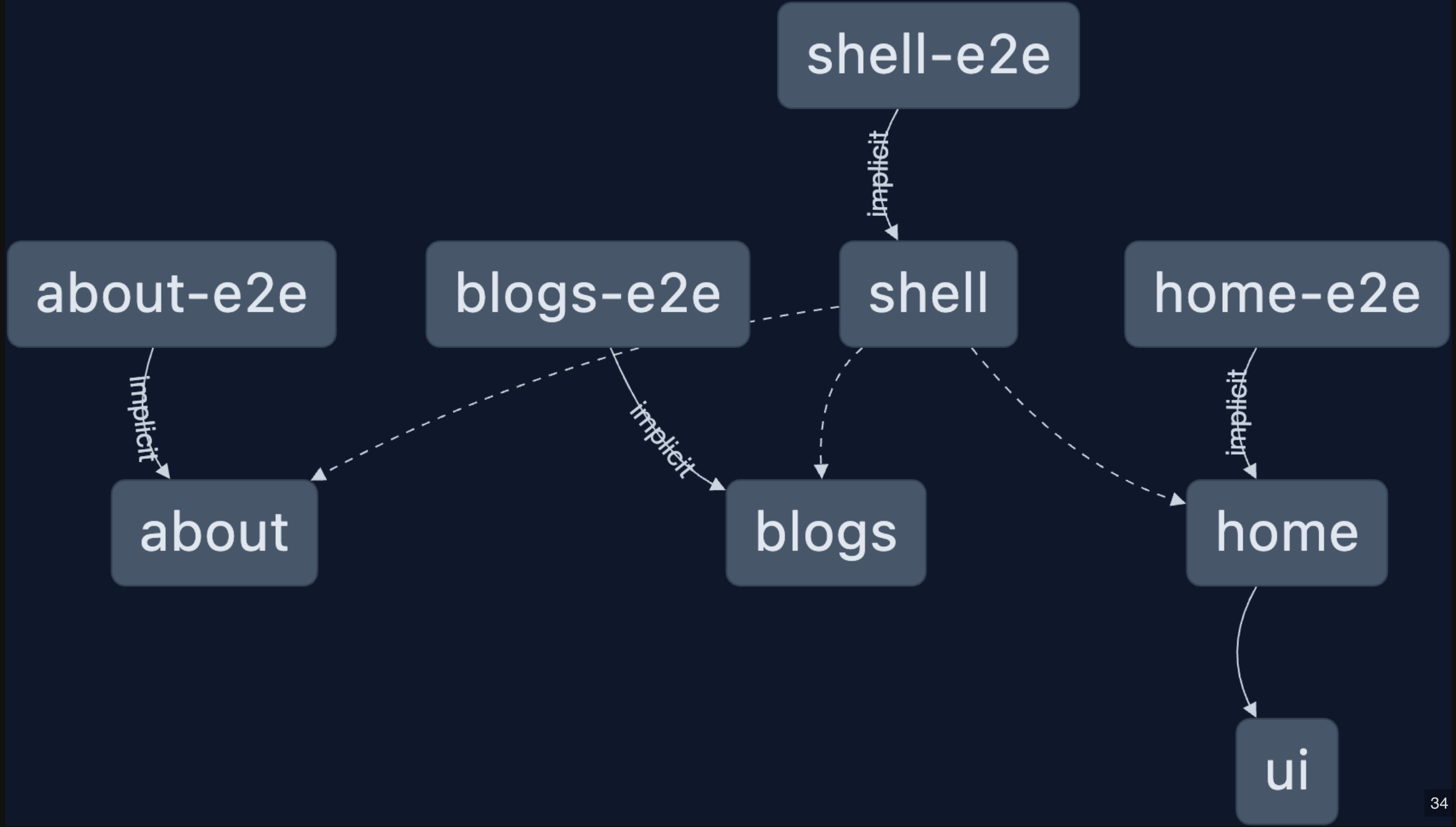
```
export default async function (tree: Tree, schema: ApplicationGeneratorSchema) {

// add files to new project if needed
generateFiles(tree, join(__dirname, 'files'), `${projectConfig.root}`, {
      name: options.name, // Pass options which can be used inside file
    });

}

// index.html.template
<h1>
<<%= name %>>
</<%= name %>>
</h1>

// Ths above file will be created with name passed as parameter on line no:5
```

Maintaining Large Codebase

Once all is removed that can be removed, that is how designs are truly in their simplest form.

1. Refactoring Code
2. Deprecations
3. Migrating code
4. Adding linters and tools
5. Helping team members
6. Documentation
7. Ability to upgrade framework version for everyone

```
1  nx graph
```


shell-e2e
implicit
about-e2e
blogs-e2e
shell
home-e2e
implicit
implicit
implicit
about
blogs
home
ui

```
nx import
  [sourceRepository]
  [destinationDirectory]
```

- Deploy documents in a central place; we use backstage.
- We use Slack to communicate deprecations and new initiatives with all developers.
- Keep a dedicated channel for developers to ask questions.
- Educate by organizing workshops.

Tools

It's always a challenge to introduce tools into a project.

- Introducing a tool into a codebase is time-consuming.
- Sometimes adding a tool is easy, but maintaining it is expensive.
- Nx makes it easy to introduce and maintain a tool.

Add e2e framework

```
nx add @nx/cypress

nx add @nx/playwright
```

Add Unit Test Tool

```
nx add @nx/jest

nx add @nx/vite:test
```

Keep your monorepo and tooling updated

```
1  nx migrate latest
```

CI/CD

We need to build 2M lines of code, and we get 100+ Pull Requests every day, we need to run Build, Unit Test, e2e.


Bernie
I am once again asking
for faster builds and tests

GitHub Actions + Nx

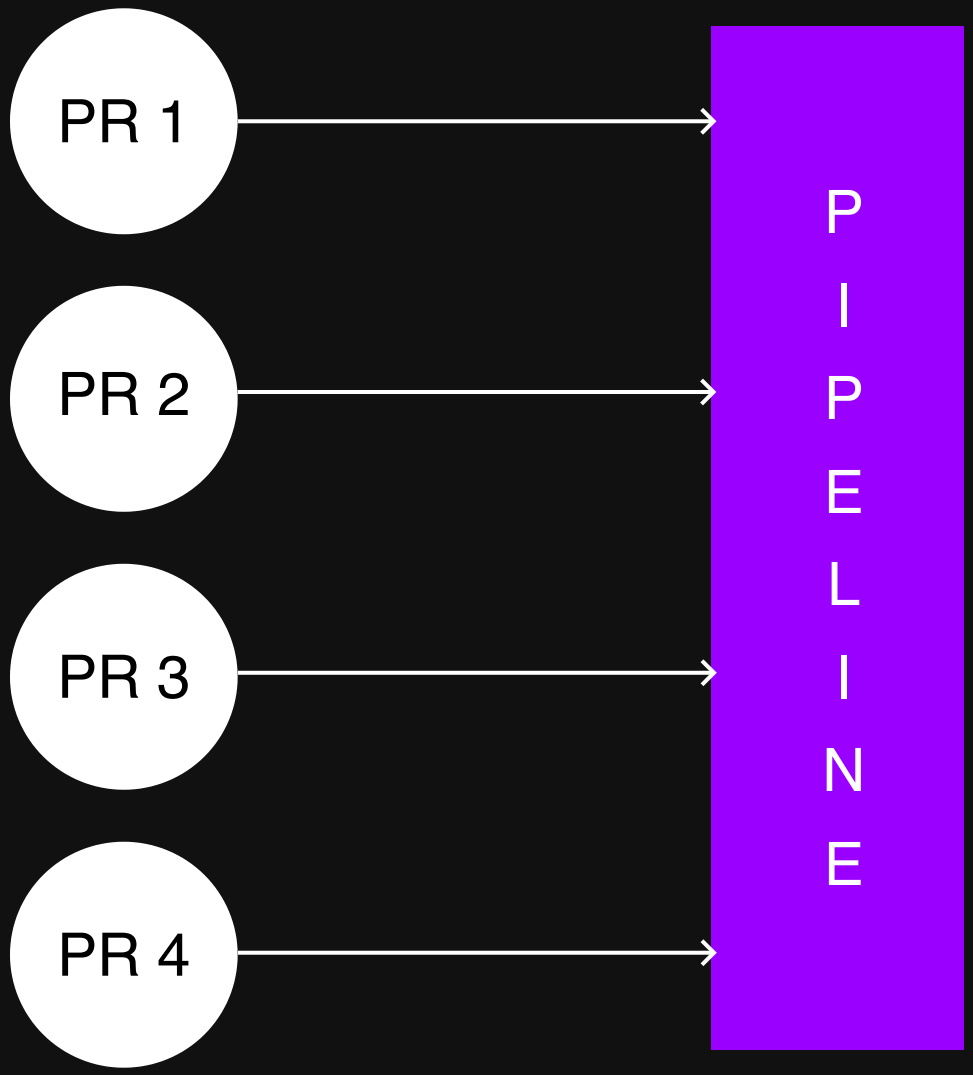
=

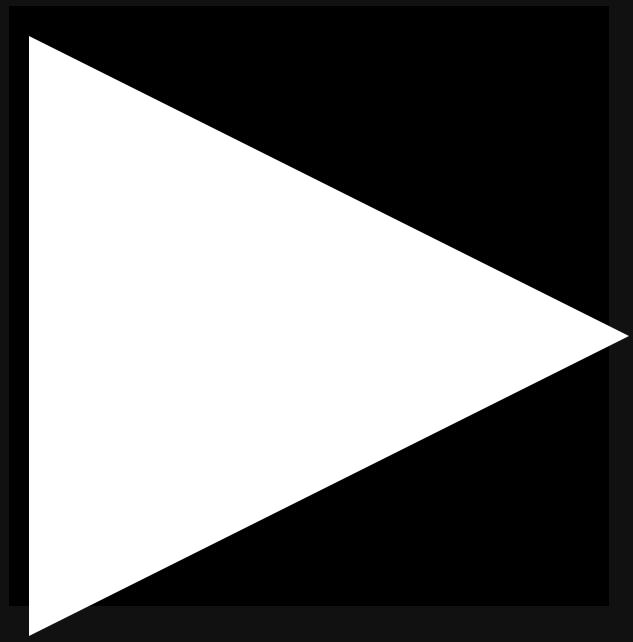
Superpower

- Use Large Runners on GitHub Actions
- Use Merge Queue to run end-to-end tests
- Cache builds for faster build and tests

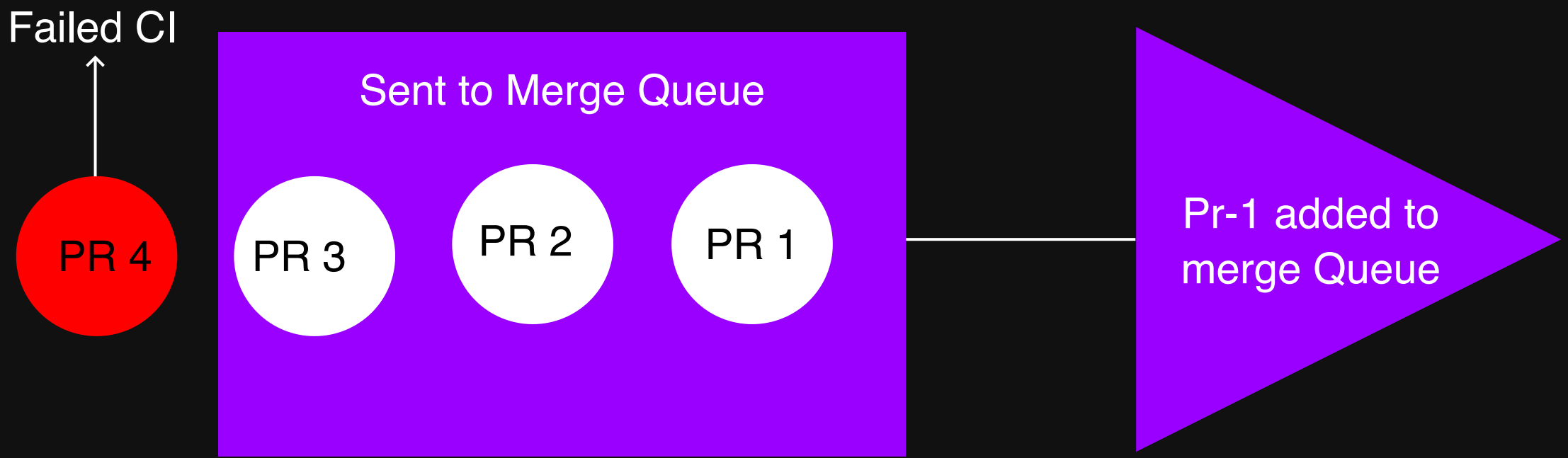
Merge Queue and User Journey Tests

- Running User Journey tests are an excellent way to avoid bugs.
- But it's costly to run on every PR.
- The branches can go out of sync when merging with the main branch.





PR = Pull Request



PR = Pull Request

PR 1

create a new branch
merge-queue/pr1

base branch is main

changes from PR 1
are merged into

merge-queue/pr1

PR = Pull Request

PR 2

create a new branch
merge-queue/pr1-pr2

base branch is
merge-queue/pr1

changes from PR 2
are merged into

merge-queue/pr1-pr2

PR = Pull Request

PR 3

create a new branch merge-queue/pr1-pr2-pr3

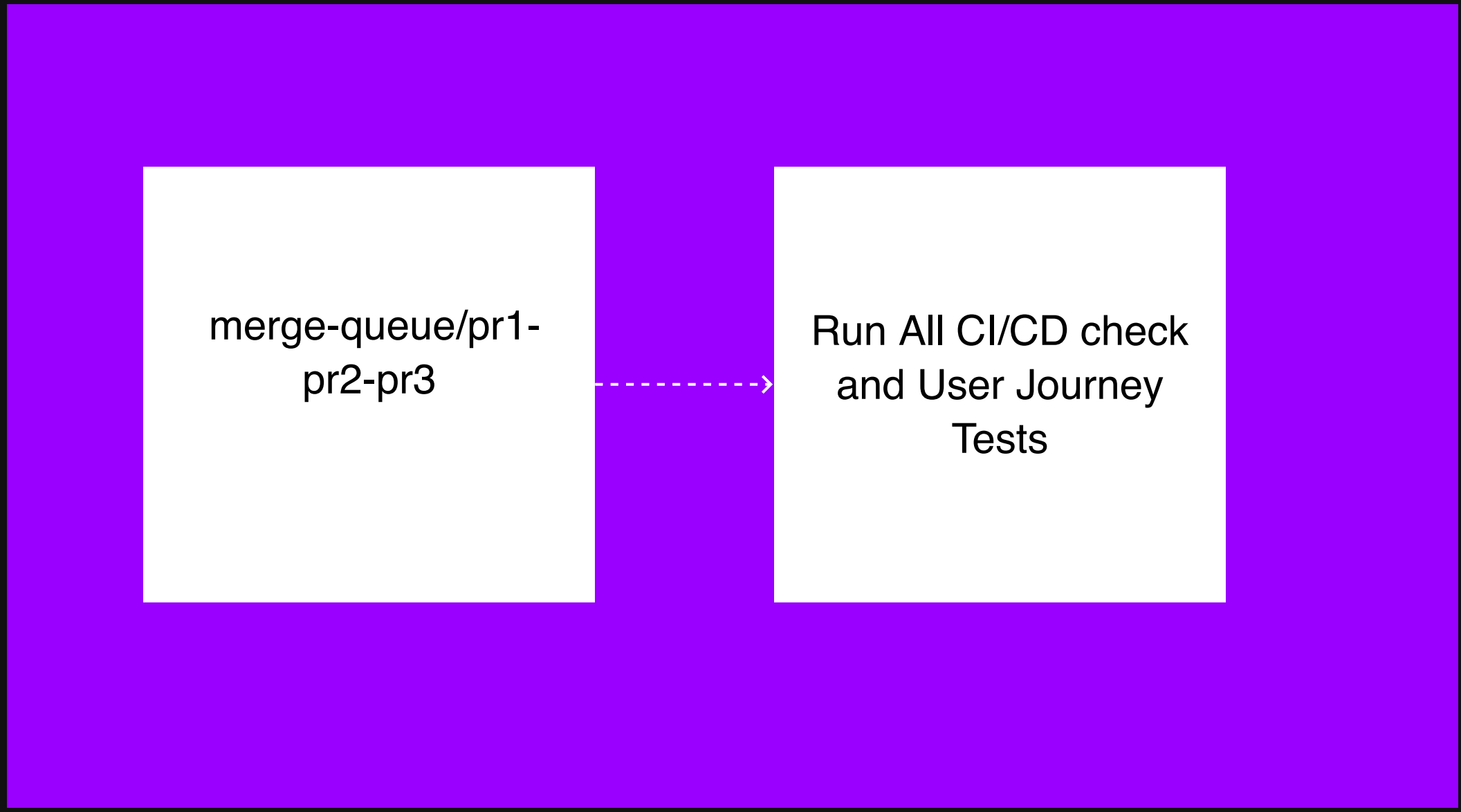
base branch is merge-queue/pr1-pr2

changes from PR 3 are merged into

merge-queue/pr1-pr2-pr3

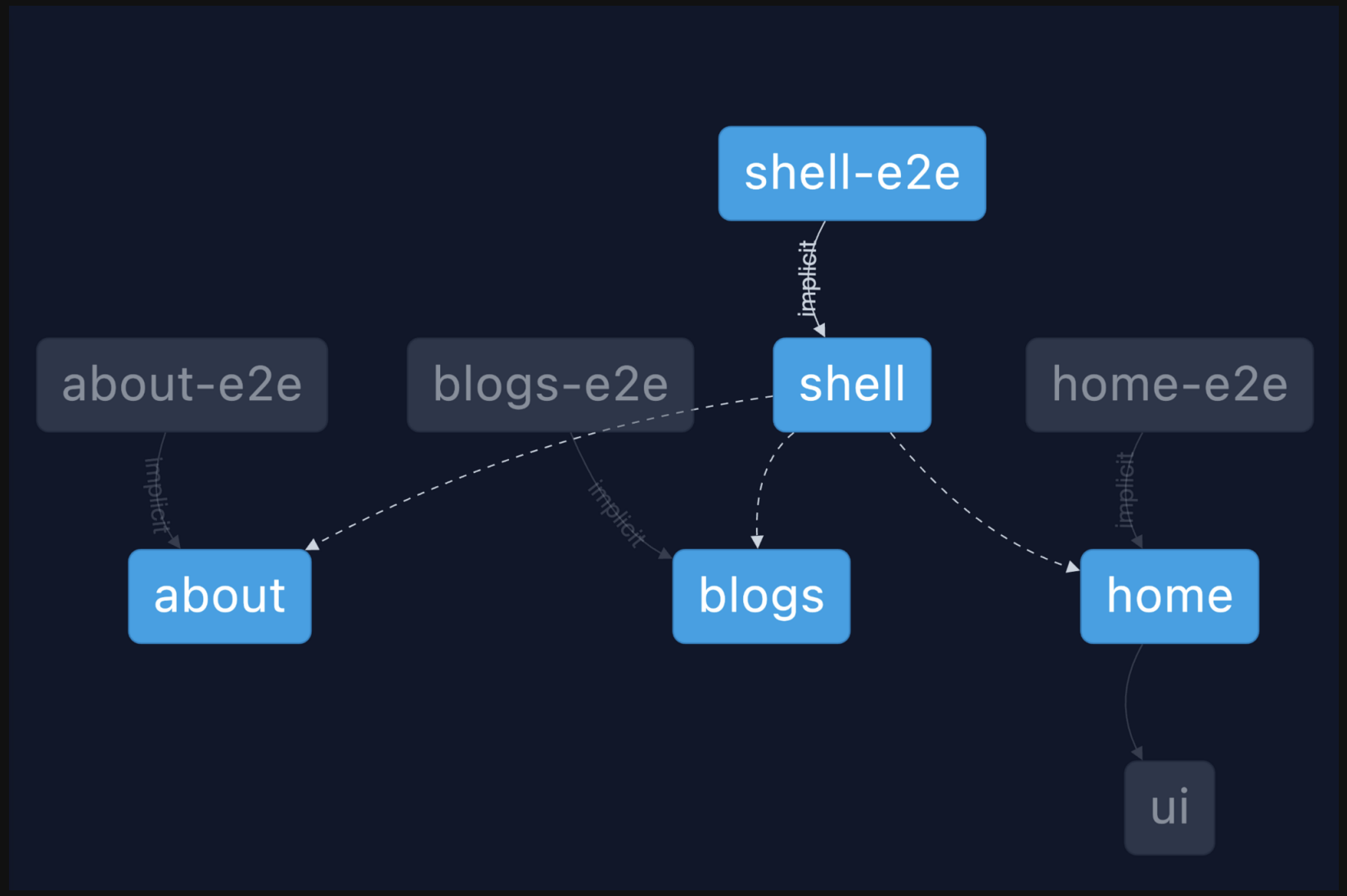
PR = Pull Request

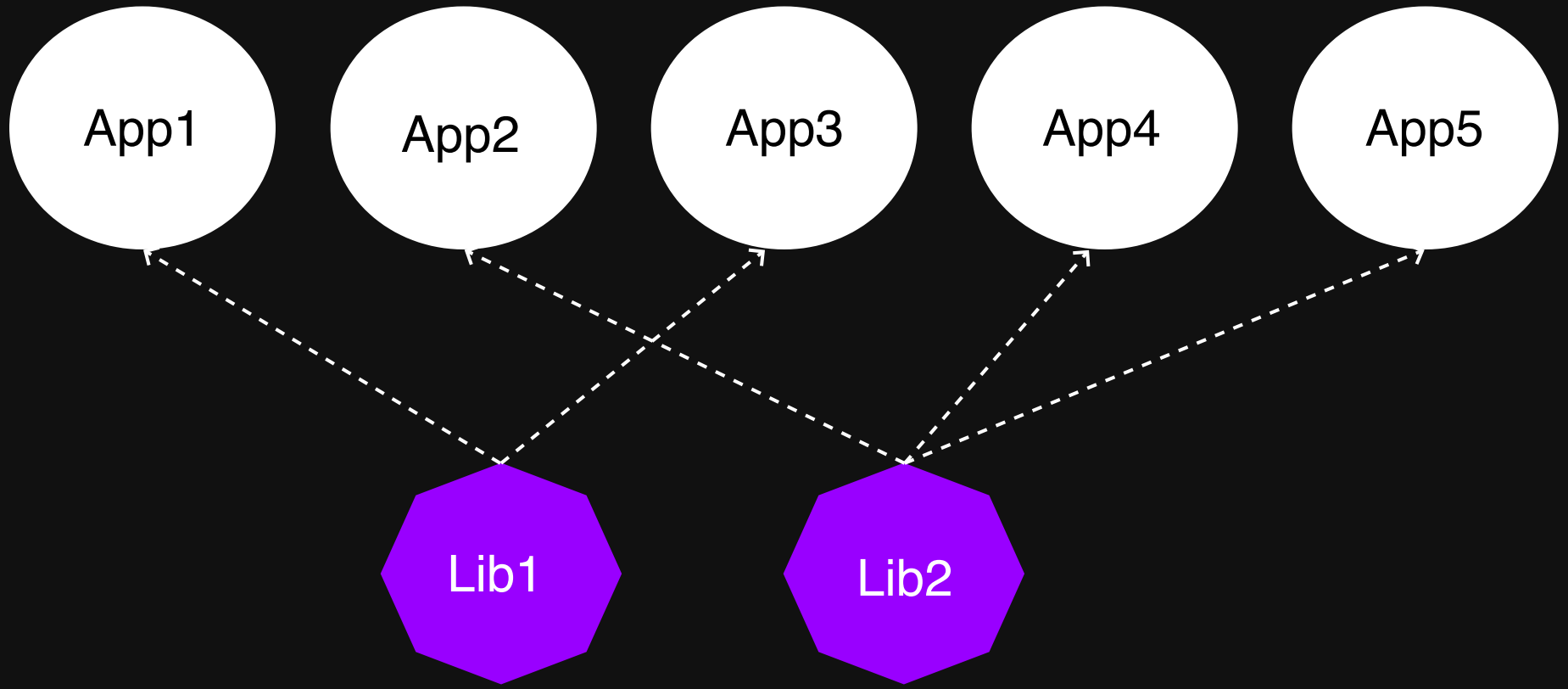
PR 3

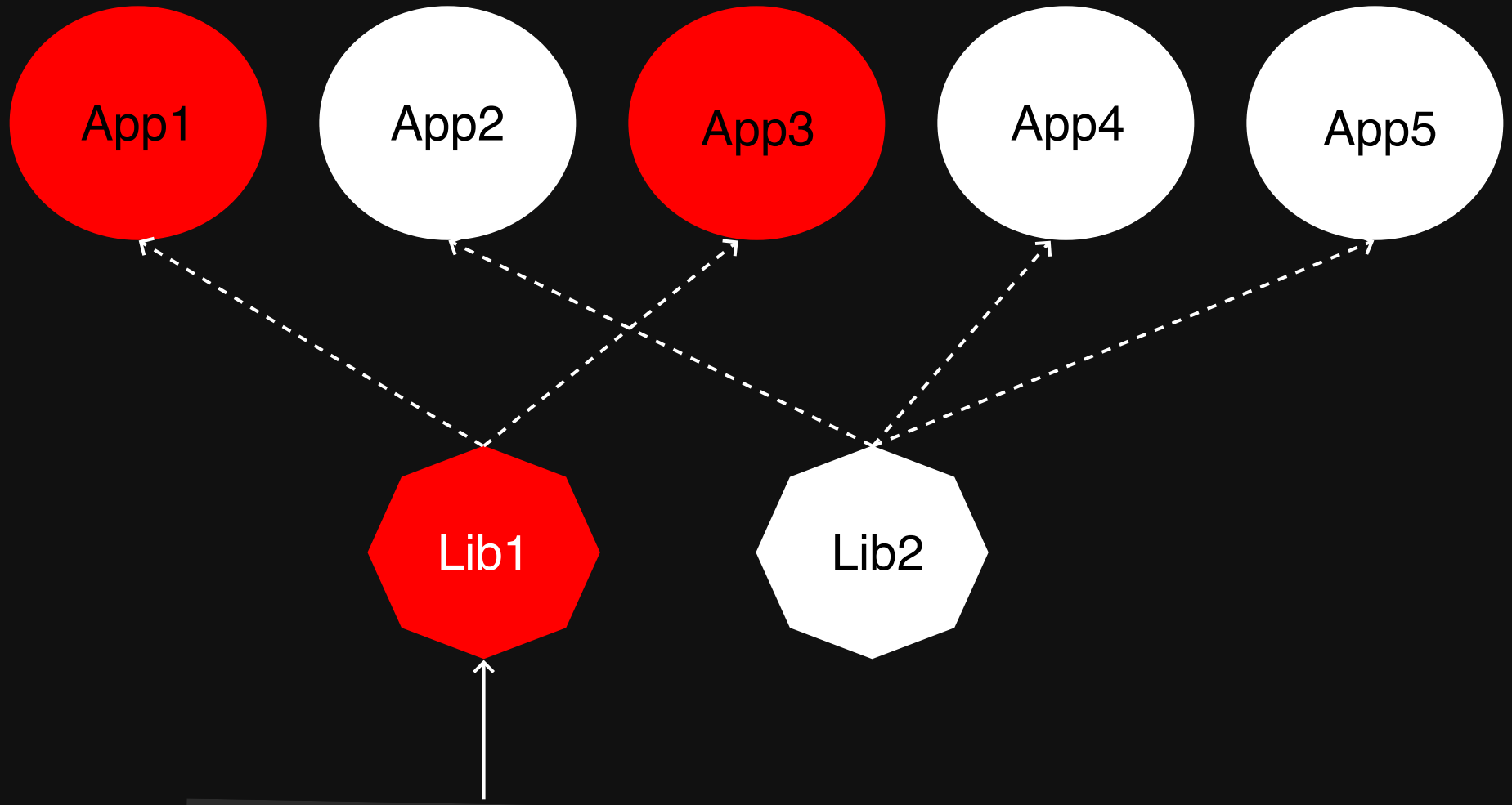


PR = Pull Request

Running Affected Tasks on GitHub




App1
App2
App3
App4
App5
Lib1
Lib2

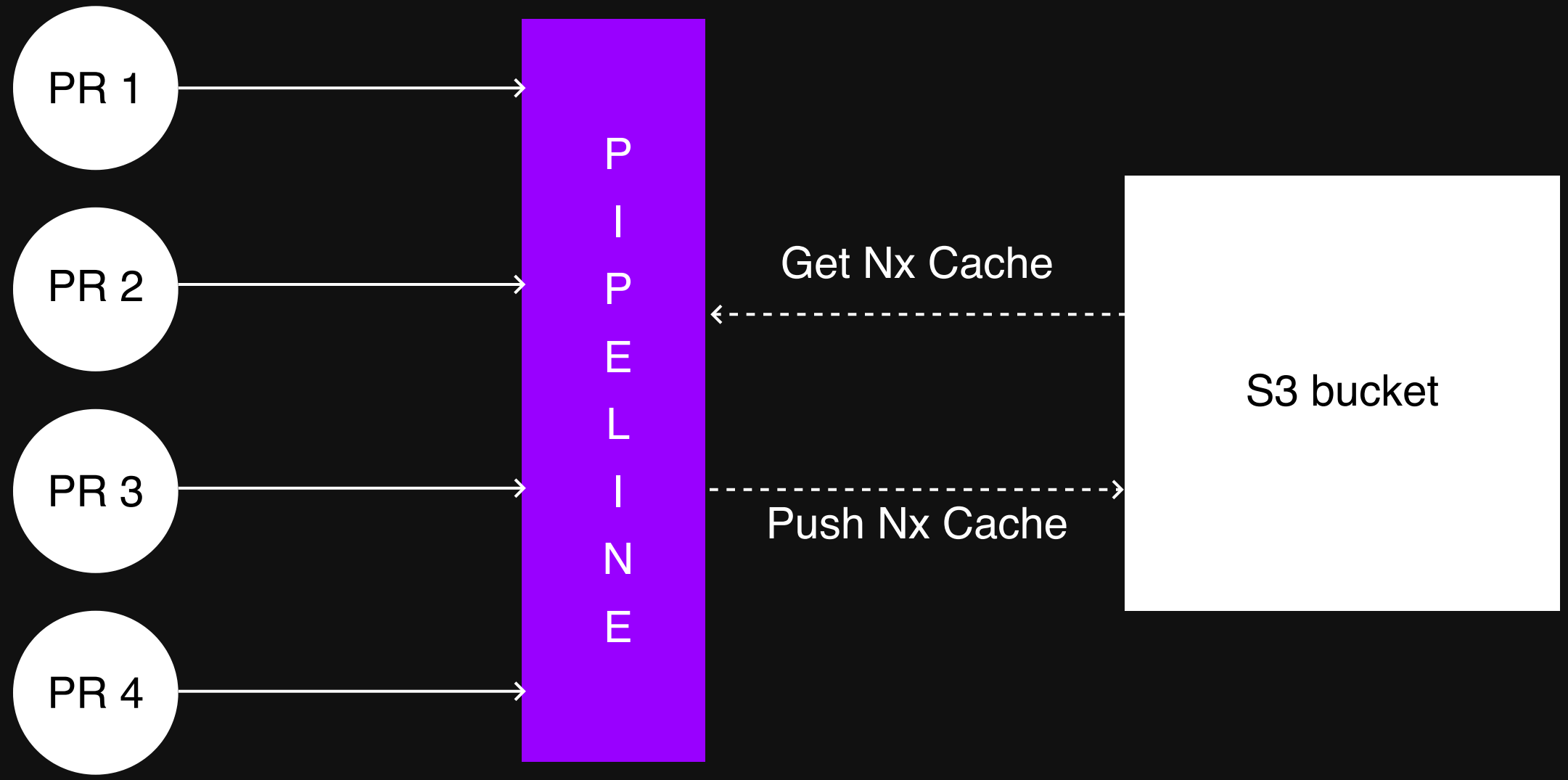



```
1 function newFeature() {
2     return true;
3 }
```

Changes in your Pull Request

Run affected tasks

```
nx affected
--targets=build,lint,test
```



PR = Pull Request

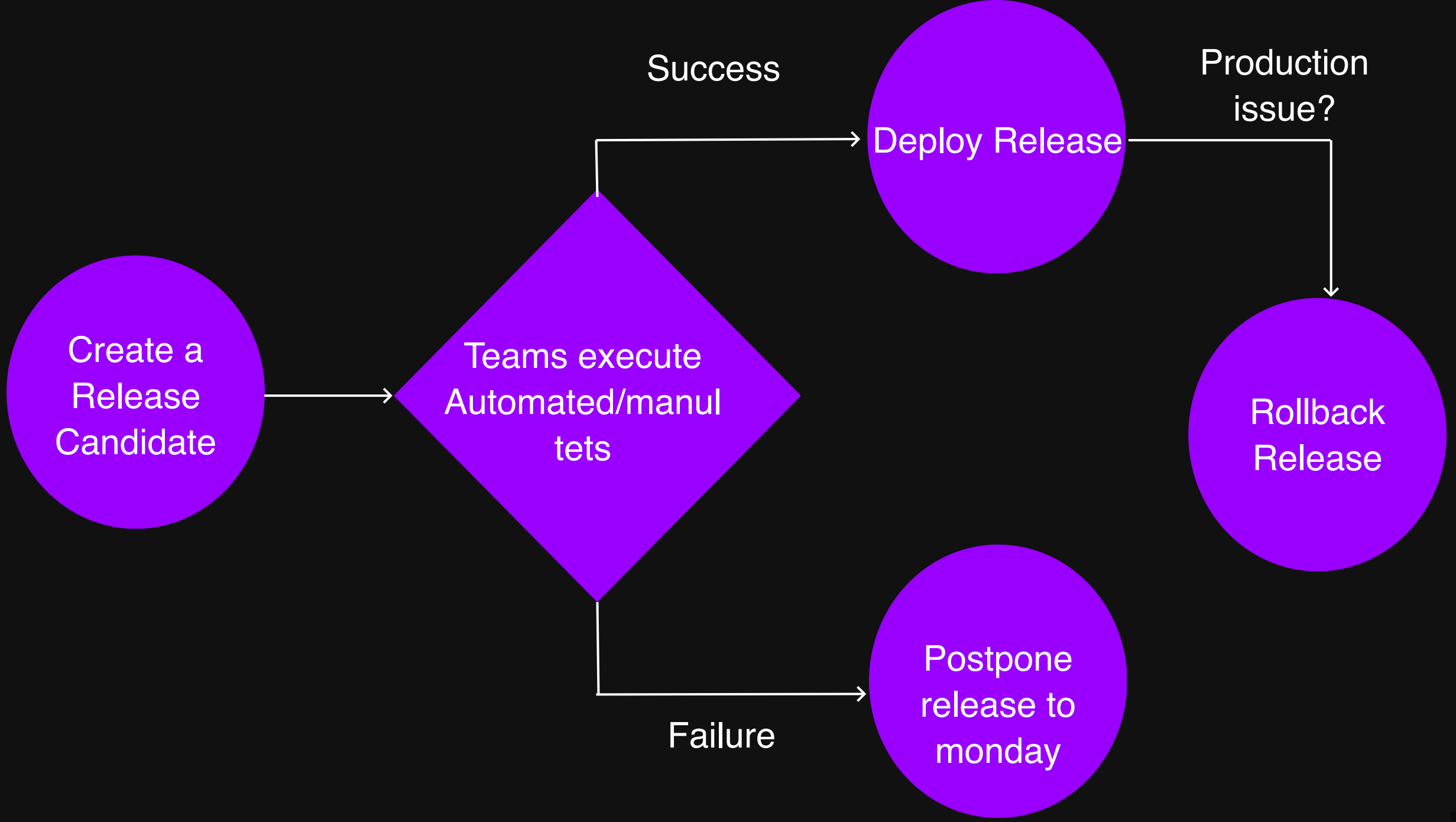

~/emotion git:(santoshyadavdev/A11Y/A11Y-386-make-sidbar-collapse-on-esc)
yarn nx run emotion:build:production
yarn run v1.22.22
warning ../package.json: No license field

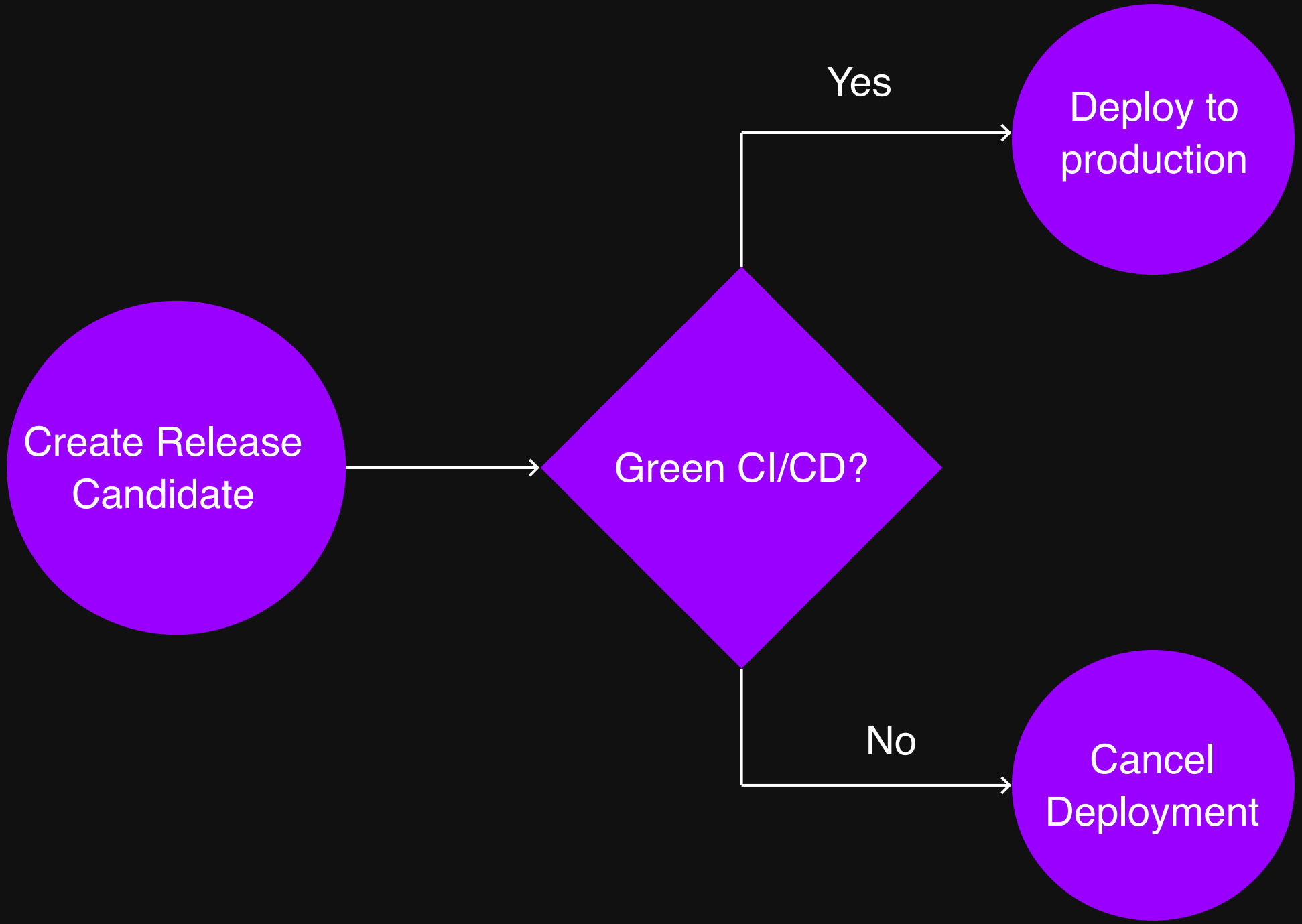

~/emotion git:(santoshyadavdev/A11Y/A11Y-386-make-sidbar-collapse-on-esc)
yarn nx run emotion:build:production
yarn run v1.22.22
warning ../package.json: No license field
$ NODE_OPTIONS=--openssl-legacy-provider node_modules/.bin/nx run emotion:build:production

Release Stratgey

Releasing 40+ apps is a challenge; we need to make sure everything is well-tested before we deploy.


DEPLOY ON A FRIDAY THEY SAID
IT'LL BE FINE THEY SAID
makeameme.org


Success
Deploy Release
Production issue?
Create a Release Candidate
Teams execute Automated/manul tets
Rollback Release
Failure
Postpone release to monday


Yes
Deploy to production
Create Release Candidate
Green CI/CD?
No
Cancel Deployment

Thank You

@SantoshYadavDev